Lab 9

Get checked off for up to 3 points of incomplete work from the previous lab within the first 10 minutes of the lab.

In this lab, you can form a group of 2-3 individuals. You must be checked off together as a group at the end of the lab. Although you perform tasks as a group, ensure that you understand the work and ask questions to TAs as needed.

Problem statement:
Towers of Hanoi is a simple puzzle that we're going to be using as practice for recursion and 2D arrays. The puzzle itself is very simple– it consists of three columns arranged from left to right, and some number of disks N of different sizes. To begin, the N disks are placed on the 1st column in order of their size, with the largest disk at the bottom of the column. The puzzle's goal is to finish with the disks arranged in the same order (biggest on the bottom, smallest on the top) on the 2nd column. Of course, you can't just move the disks however you want! You need to follow these rules:

- You can only move one disk at a time by taking it off the top of its peg and putting it onto another peg.
- You're not allowed to place a disk on top of another disk that's smaller– that is, every disk must be smaller than every disk beneath it on the peg.

You can do the simulation here: https://www.mathsisfun.com/games/towerofhanoi.html

(4 pts) Design

First, begin by writing the steps on a piece of paper that represents the moves among the columns. For instance, with three disks, the smallest disk from the 1st post will be moved to the second peg, i.e. 1 -> 2. Then, the 2nd disk will be moved to the 3rd peg, i.e. 1->3, etc.

Write the steps for the base case, n = 1 disks, n = 2 disks, and n = 3 disks. You should notice that you have 2n – 1 moves for each of these cases. Also, note any pattern that you see, i.e. when do you see the base case.

Here is an outline of the recursive towers function, feel free to add another parameter if needed, i.e. **total_disks**.:

```
void towers(int number_of_disks, int b[ ][3], int from_col, int to_col, int spare) {
        If(number of disks is >= 1)
                Call Towers with (number_of_disks-1, b, from_ col, spare, to_ col)
                Move the disk
                Print the board
                Call Towers with (number_of_disks-1, b, spare, to_ col, from_ col)
}
```

As a group with the TAs, walk through the algorithm provided for the towers() function with a board that has 1 disk and 3 columns, then 2 disks and 3 columns, and 3 disks with 3 columns, e.g. . towers(1, 1, 2, 3);, towers(2, 1, 2, 3);, towers(3, 1, 2, 3);, etc.

Provide the example walk through for the following calls:

towers(1, 1, 2, 3);
towers(2, 1, 2, 3);
towers(3, 1, 2, 3);

(6 pts) Implementation : Statically Allocated 2-D array

First, you can implement this using a static **2-D array with 3 columns for the 3 posts and 3 rows**, and you can initialize the array with the numbers 1, 2, and 3 in the first column to represent the initial state of the game. The goal is to print out the board after each move in the game, seeing the following output. **Example with two disks**:

```
1 0 0
2 0 0
----------
0 0 0
2 0 1
----------
0 0 0
0 2 1
----------
0 1 0
0 2 0
----------
```

The above implementation can also be done using a dynamically allocated array. Let us figure it out in next lab

Show your completed work and answers to the TAs for credit. You will not get points if you do not get checked off!